AO Middleware Supporting Variability and Dynamic Customization of Security Extensions in the ORB Layer

Stefan Walraven & Petrus Verbaeten
Dept. of Computer Science
K.U.Leuven, Belgium
{firstname.lastname}@cs.kuleuven.be

ABSTRACT

There is an increasing demand for variability and dynamic customization of middleware. Security, a key middleware service, requires variations at design-time, deploy-time as well as runtime. This is hard to achieve with state-of-the-art middleware.

To handle this problem, we applied an SPL approach based on static and dynamic AO-composition in a component-based middleware platform. As proof of concept we developed a prototype with several variations in the security domain. A first evaluation shows that AO and component-based middleware are promising technologies to obtain an SPL.

Categories and Subject Descriptors

C.2.4 [Computer-Communication Networks]: Distributed Systems; D.2.11 [Software Engineering]: Software Architectures

General Terms

Design, Management, Security

Keywords

Middleware, Variability, Customizability, Aspect-Orientation, Security

1. PROBLEM CONTEXT

State-of-the-art middleware for distributed applications faces an increasing demand for flexibility as well as an increasing heterogeneity of the environment. Basic middleware platforms provide a set of common services that are often used in distributed applications. However, there is a need for variability and customization of middleware, driven by requirements of the target domain, the target deployment environment, hosted applications as well as the end-users of the distributed system.

The built-in security services are the key middleware services that face all of these variation requirements. Today's middleware platforms include design-time variations of ORBs such as basic ORBs for intranet communication as well as more advanced ORBs with integrated security protocols for authentication, integrity and encrypted communication. Deployment environments may require specific security extensions to the middleware platform to support specific authentication technologies. For example, banks integrate in-house authentication mechanisms for their employees. Money transfers and cash withdrawals in a retail banking software application require application-specific access rules with regard to transfer limits and accumulative cash withdrawal limits. A typical example of user-driven customization is the selection of the authentication mechanism when logging in into the web banking environment.

Variations of middleware platforms thus require extensions of the basic middleware at middleware design-time, middleware deploy-time, application deploy-time as well as application runtime. These extensions are crosscutting through the distributed software and also affect the deeper layers of the middleware. Current component- and aspect-based middleware fails to support *all* these extensions.

2. GOAL & APPROACH

At the level of functionality, security features in middleware are specified as an *extension*, composed with the basic functionality of the middleware platform. This (de)composition model should be reflected in the design and the implementation. A software product line (SPL) approach handles this problem. Our composition technology to support such an SPL approach is based on static and dynamic AO-composition in a component-based middleware platform.

A *software product line* is a collection of software systems satisfying the specific needs of a particular domain, that share a common set of *core assets* in a prescribed way. SPL enables developers and architects to manage the variability between products and to deliver middleware platforms tailored to the requirements of the user or device. The core asset of our middleware product line, is a minimal ORB architecture that forms a stable base for extension. This core distribution layer is composed with the desired security extensions at the moment they are needed: from design-time until application runtime. Supporting middleware security

Figure 1: Different variations of the core distribution layer

services as extensions to such a stable base requires a component and composition technology that can offer modularity of the stable base and its extensions as well as advanced, powerful composition techniques to tackle the *crosscutting* and *deep* compositions. Component-based middleware and AO middleware have already shown promising results in handling deep, respectively crosscutting composition in middleware.

A *component-based design* of the middleware platform has been proven useful to achieve openness and adaptability of services in the platform. In such component-based middleware, the components are middleware services, that can be customized, configured and composed with other middleware services. An example of such a middleware is OpenCOM [1] and the work by Jørgensen et al. [3]. However, customization through replacement of components in the basic middleware requires a lot of reimplementations of the functionality of the original component. For example, the use of a secure transport layer means often a complete replacement of the original transport layer.

Aspect-orientation (AO) [4] tackles the interdependencies between components by focusing on the identification, representation and composition of often crosscutting concerns throughout the software development process. Aspect-oriented middleware has been put forward as a promising and relatively successful paradigm to improve modularity and customization capabilities of middleware platforms, but also for modular extension with new crosscutting services [5, 2, 6]. However, these approaches typically only offer an ad-hoc solution for one of the customization problems and cannot cope with the complete set of requirements of an SPL: variations at design-time, deploy-time and runtime.

3. ARCHITECTURE & PROTOTYPE

The core asset of the SPL is a component-based ORB, which is the stable base for adding security extensions. It consists of a transport layer, a marshalling layer and a proxy - skeleton (see Figure 2). This core ORB offers only basic facilities like multithreading and sessions. It provides interfaces on which pointcuts can be defined and specialized implementations can be bound. The setup of the basic ORB occurs at runtime.

The ORB is underpinned by an AO-kernel. This kernel uses an aspect-component model with advanced static and dy-

namic AOP support and ensures thread safety to support AO-composition in the ORB.

With AO-composition one can create different variations of the distribution layer, at design-time, deploy-time or run-time. For our prototype we developed two variations in the security domain: a variation that uses SSL to secure the communication and one that requires authentication at an authentication server (AS). Both these variations use aspects that operate on the interfaces of the transport layer (see Figure 1).

Figure 2: The core ORB with authentication extension

Another variation we implemented provides the basic ORB with automatic protocol selection (APS). Servers that use this variation will be capable to support multiple technologies, e.g. different security protocols. Clients choose the protocol based on what is supported or desired. The automatic protocol selection gives rise to even more variations: every remote component can have its own list of supported technologies. The automatic protocol selection makes use of dynamic AO-composition to realize “just-in-time activation” of the chosen protocol.

4. CONCLUSION AND FUTURE WORK

Variability and dynamic extensions are hard to achieve in state-of-the-art middleware. We have developed a minimal ORB which can be fully and dynamically composed and extended by the systematic use of aspects. Security extensions can be achieved at design-time, deploy-time and runtime.

This way we have handled both the problem of limited flexibility and the demand for SPL's.

Our architecture has been validated in a prototype and shows that AO and component-based middleware are promising technologies to obtain an SPL. As further validation and evaluation are necessary, we plan to extend our prototype with other crosscutting services that affect the deeper layers, for example transaction and threading support.

5. REFERENCES

- [1] CLARKE, M., BLAIR, G. S., COULSON, G., AND PARLAVANTZAS, N. An efficient component model for the construction of adaptive middleware. In *Middleware '01*, Springer-Verlag, pp. 160–178.
- [2] COLYER, A., AND CLEMENT, A. Large-scale AOSD for middleware. In *AOSD '04*, ACM, pp. 56–65.
- [3] JØRGENSEN, B. N., TRUYEN, E., MATTHIJS, F., AND JOOSEN, W. Customization of object request brokers by application specific policies. In *Middleware '00*, Springer-Verlag New York, Inc., pp. 144–163.
- [4] KICZALES, G., LAMPING, J., MENDHEKAR, A., MAEDA, C., LOPES, C. V., LOINGTIER, J.-M., AND IRWIN, J. Aspect-oriented programming. In *Proceedings ECOOP (1997)*.
- [5] PAWLAK, R., DUCHIEN, L., FLORIN, G., LEGOND-AUBRY, F., SEINTURIER, L., AND MARTELLI, L. JAC: an aspect-based distributed dynamic framework. *Software-Practice & Experience* 34(12) (2002), 1119–1148.
- [6] ZHANG, C., GAO, D., AND JACOBSEN, H.-A. Towards just-in-time middleware architectures. In *AOSD '05*, ACM, pp. 63–74.